REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
FEBRUARY 1994

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
Software Reengineering Project Planning Guide, Version 2.0.

5. FUNDING NUMBERS

6. AUTHOR(S)
Ms Tamra Moore, editor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DISA/JIEO/ CFSW
701 South Courthouse Road
Arlington, Va 22204-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
same as above

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Field 25 should contain the identifier CIM (Collection), as detailed in A. Washington DTIC-OCS IOM, dated April 11, 1994.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Available for public distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document assists project managers in developing a plan to implement software reengineering for AISs. Developing a solid project plan is the first step of any software engineering process, including software reengineering. The intended audience for the guide is any organization within DoD tasked to reengineering AISs.

14. SUBJECT TERMS
reengineering, reverse engineering, software engineering, maintenance, IDEF

15. NUMBER OF PAGES
37

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
unclassified

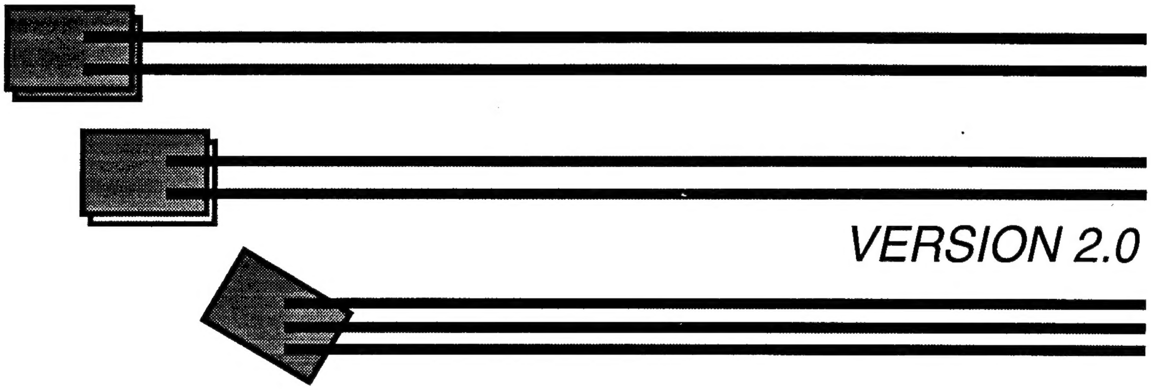
18. SECURITY CLASSIFICATION OF THIS PAGE
unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
unclassified

20. LIMITATION OF ABSTRACT

Defense Information Systems Agency
Joint Interoperability and Engineering Organization
Center for Software
701 South Courthouse Road; Arlington, VA 22204-2199

Software Reengineering Project Planning Guide



VERSION 2.0

February 1994

Prepared by:

Software Systems Engineering Department
Software Engineering Technology Division

FOREWORD

1. The Software Reengineering Planning Guide is intended for use by all Departments and Agencies of the Department of Defense. This Guide assists project managers in developing a plan to implement software reengineering for automated information systems (AISs). Developing a project plan is the first step of any software engineering process, including software reengineering. A software reengineering process is defined in the Software Systems Reengineering Process Model [CFSW94]. This process begins with project planning described in the activity called Define Project. The Software Reengineering Project Planning Guide provides guidance for performing tasks described in the *Define Project* activity.

2. Beneficial comments (recommendations, additions, deletions) and any pertinent data which may be of use in improving this document should be addressed to:

Defense Information Systems Agency
Joint Interoperability and Engineering Organization
Center for Software
Software Systems Engineering Department
701 South Courthouse Road, Arlington, VA 22204-2199

3. The DoD Components may obtain copies of this document through their own publication channels. Defense Contractors, and other Federal Agencies may obtain copies from:

Defense Technical Information Center (DTIC)
Building 5 Cameron Station
Alexandria, VA 22304-4301
Commercial Telephone: 1-800-225-DTIC (1-800-225-3842)

4. This Guide is not intended to specify or discourage the use of any particular software engineering project planning method. The Guide adapts current project management planning strategies for software engineering to provide a format and define the contents of the software reengineering project plan.

5. The intended audience for the Software Reengineering Project Planning Guide is any organization within DoD tasked to reengineer AISs. This Guide must be appropriately tailored by the project manager to ensure that the necessary and cost-effective activities of software reengineering are implemented. Assistance in tailoring this document is available from the Software Reengineering Program located at the address in paragraph 2 of this Foreword.

6. The Center for Software[1] is chartered to support the Office of the Assistant Secretary of Defense (OASD) for Command, Control, Communications, and Intelligence (C^3I) by providing information management technical services to the DoD community. The services are an integral part of the Corporate Information Management program, a DoD-wide effort to streamline business operations and processes which will help improve the design of cost-effective, standard information systems. Software reengineering emerges as a strategy for bringing the cost of developing and maintaining software under control. The need for a comprehensive plan to apply software reengineering technology is the driving force behind the Software Reengineering Program. The Software Reengineering Project Planning Guide will assist managers in planning and implementing software reengineering technology.

[1] The Center for Software includes the organization formerly named the Center for Information Management in the Defense Information Systems Agency's Joint Interoperability and Engineering Organization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

(This page was intentionally left blank.)

1. INTRODUCTION

1.1 Scope. This document provides a format and describes the contents of software reengineering project plans. A project plan is the controlling document for managing software reengineering projects by defining both technical and managerial processes for accomplishing the goals of the project. This document utilizes proposed software engineering project planning strategies for outlining the structure and describing the contents of project plans for software reengineering applications. Sources for these strategies are listed in the References section of this document.

This document can be used to plan any project utilizing software reengineering technology. The main purpose of this document is to facilitate the software reengineering process as defined in the Software Systems Reengineering Process Model [CFSW94]. This process begins with initial project planning as described in the activity called *Define Project* (Figure 1). The primary product of the Define Project activity is the Reengineering Project Plan. Define project is composed of three high-level activities, including *Define Objectives*, *Identify Baseline*, and *Define Reengineering Project Plan* (Figure 2). The Define Reengineering Project Plan activity is performed in four parts, including *Develop Reengineering Strategy*, *Identify Methodologies and Tools*, *Allocate Resources*, and *Produce Reengineering Project Plan* (Figure 3). Chapter Three describes the relationship to the activities in the Software Systems Process Model to the structure and contents of the project plan.

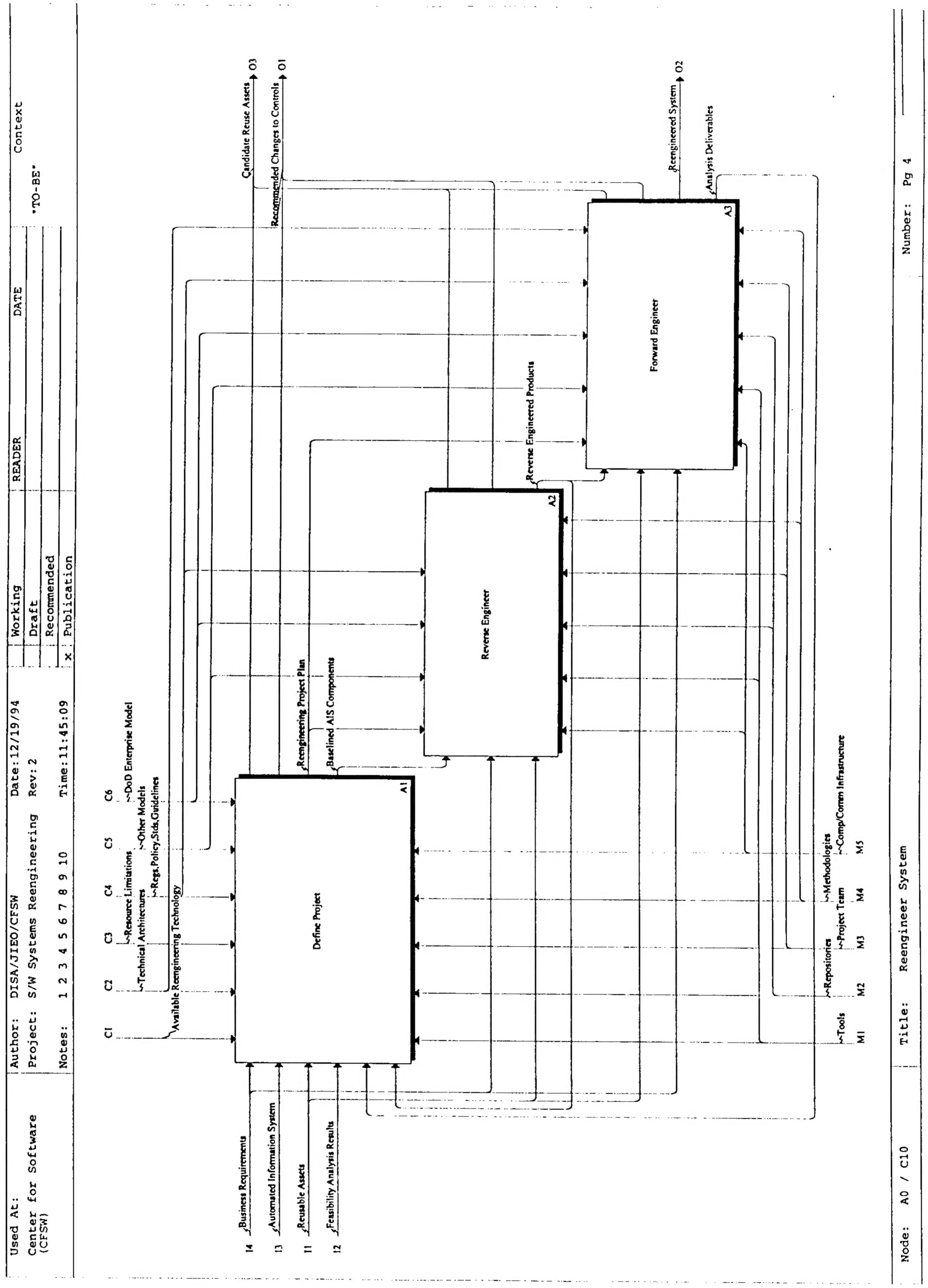


Figure 1. Reengineer System

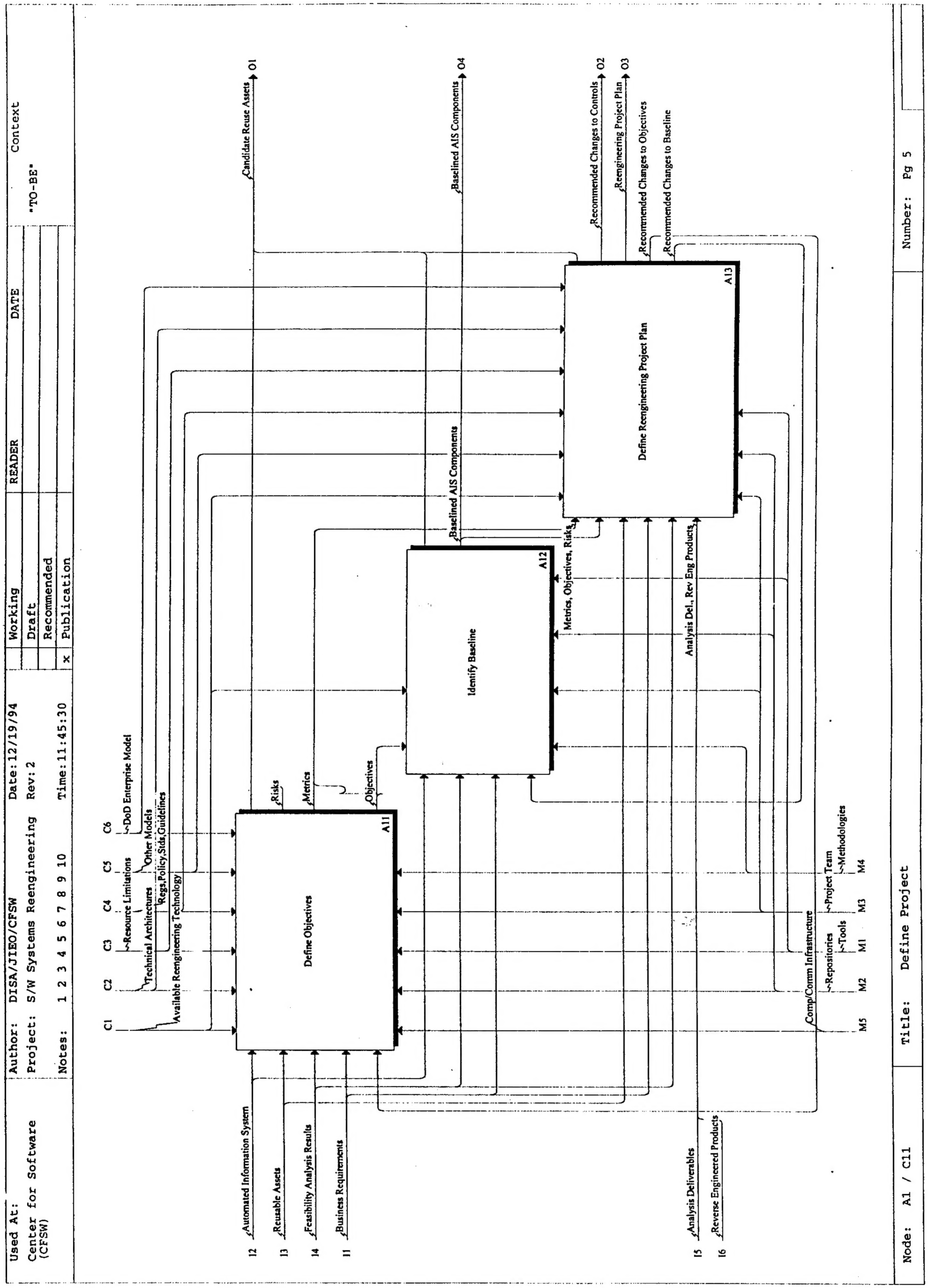


Figure 2. Define Project

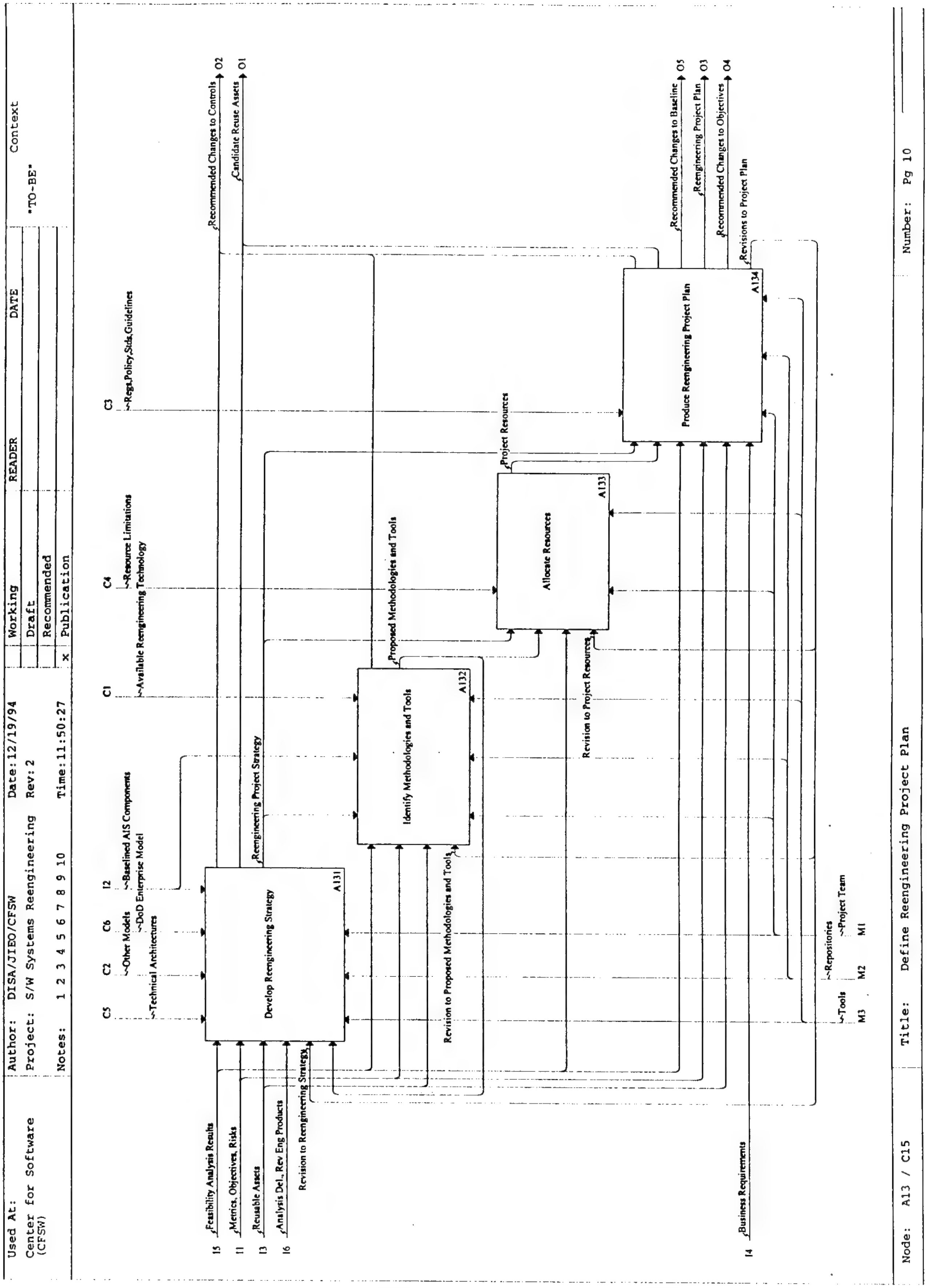

Used At:
Center for Software (CFSW)
Author: DISA/JIEO/CFSW
Project: S/W Systems Reengineering
Notes: 1 2 3 4 5 6 7 8 9 10
Date: 12/19/94
Rev: 2
Time: 11:50:27
Working
Draft
Recommended
x Publication
READER
DATE
Context
"TO-BE"
C5
C2
C6
I2
C1
C4
C3
Technical Architectures
Other Models
DoD Enterprise Model
Baselined AIS Components
Available Reengineering Technology
Resource Limitations
Regs,Policy,Stds,Guidelines
Develop Reengineering Strategy
A131
Identify Methodologies and Tools
A132
Allocate Resources
A133
Produce Reengineering Project Plan
A134
Reengineering Project Strategy
Proposed Methodologies and Tools
Project Resources
Revision to Proposed Methodologies and Tools
Revision to Project Resources
Recommended Changes to Controls
O2
Candidate Reuse Assets
O1
Recommended Changes to Baseline
O5
Reengineering Project Plan
O3
Recommended Changes to Objectives
O4
Revisions to Project Plan
I5
Feasibility Analysis Results
I1
Metrics, Objectives, Risks
I3
Reusable Assets
I6
Analysis Del., Rev Eng Products
Revision to Reengineering Strategy
I4
Business Requirements
Tools
M3
Repositories
M2
Project Team
M1
Node: A13 / C15
Title: Define Reengineering Project Plan
Number: Pg 10

Figure 3. Define Reengineering Project Plan

This document provides a format for the software reengineering project plan. Appendix A includes templates for developing a project plan based on the guidance provided in this document. Individual DoD organizations may have a standard format for all project plans. The reader should use this as a guide for planning software reengineering efforts, and should incorporate the guidance provided in this document into existing organizational standards concerning the structure and format of project plans.

This document describes the minimal set of elements which should appear in a software reengineering project plan, but the content of the plan is not limited to these elements. Individual organizations and projects should use this as a guide for planning software reengineering efforts, and should investigate additional elements for inclusion in the plan as required by these organizations.

Methodologies and commercial products are available to assist project managers in automating many of the activities described in this planning guide. Configuration management and project planning tools should be investigated for automating and documenting the work during this activity.

1.2 Standards. The following government and industry standards should be consulted when applying the guide. The latest revisions apply.

[1] DoD-STD-1703(NS), Software Product Standards.

[2] DoD-STD-2167A, Defense System Software Development.

[3] MIL-STD-7935A, DoD AIS Documentation Standards.

[4] MIL-STD-SDD, Military Standard for Software Development and Documentation.

[5] ANSI/IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology.

[6] ANSI/IEEE Std 1058.1-1987, IEEE Standard for Software Project Management Plans.

[7] ANSI/IEEE Std. 1061-1992, Standard for a Software Quality Metrics Method

1.3 Overview. The reengineering project plan has the three main objectives of software engineering project planning: to (1) communicate the scope and resources; (2) define cost and schedule; and (3) define the overall approach [Pres87, p131]. The reengineering project plan provides guidance on how the project is progressing, what resources are available and when these resources will be used.

General principles for management apply in software reengineering projects. The following general management principles were adapted from [Blum92, p429]:

Understand the goals and objectives for the reengineering effort
Understand the constraints which could impact the reengineering effort
Plan to meet objectives within these constraints
Monitor and maintain the project plan
Make adjustments as necessary
Preserve calm, productive, and positive work environment

The reengineering project plan documents the information supporting these principles. The plan outlines what has to be done (the activities) and the best order for these activities to be performed. Short term, clear objectives will motivate personnel [Youl90, p18]. Well-defined, intermediate tasks will assist the project manager in determining how well the project is progressing.

The project plan documents estimated costs, including staff, training, and computer resources. The plan identifies the number and skills of personnel needed. Training on methodologies and tools for software reengineering, as well as modern software engineering should be provided for all personnel. Resources are identified based on availability and cost. Personnel should have access to these resources to perform the software reengineering.

The expected completion date of the software reengineering project is projected in the plan, including interim milestones and associated products. The completion of these milestones and the quality of the products produced may indicate how the project is progressing.

The primary elements of a software reengineering project listed below are adapted from Whitten's software engineering elements [Whit90, p5]:

people:	discipline communicating education	processes:	schedules quality tracking priorities	activities:	product objectives training specifications ease of use informal testing

Figure 4. Primary Elements of Software Reengineering

Successful planning for software reengineering manages these key elements. The Software Reengineering Project Planning Guide assists project managers in establishing a plan for managing these elements and achieving the goals of the project.

2. DEFINITIONS

A set of definitions are provided for clarification. These terms are used throughout the document and are defined as follows. These definitions are provided by the ANSI/IEEE Std 1058.1-1987, IEEE Standard for Software Project Management Plans unless otherwise stated.

activity: a major unit of work to be completed in achieving the objectives of a software project. An activity has precise starting and ending dates, incorporates a set of tasks to be completed, consumes resources, and results in work products. An activity may contain other activities in a hierarchical manner.

baseline: A work product that has been formally reviewed and agreed upon, and that can be changed only through formal change control procedures. A baseline work product may form the basis for further work activities.

legacy system: Other automated information systems (AISs) that duplicate the support services provided by the migration system, and are to be terminated [DSD93].

migration system: An existing AIS, or a planned and approved AIS, that has been officially designated as the single AIS to support standard processes for a function [DSD93].

software project management plan: The controlling document for managing a software project. A software project management plan defines the technical and managerial project functions, activities, and tasks necessary to satisfy the requirements of a software project, as defined in the project agreement.

task: The smallest unit of work subject to managerial accountability. A task is a well-defined work assignment for one or more project members. The specification of work to be accomplished in completing a task is documented in a work package. Related tasks are usually grouped to form activities.

work package: A specification for the work to be accomplished in completing an activity or task. A work package defines the work product(s), the staffing requirements, the expected duration, the resources to be used, the acceptance criteria for the work products, the name of the responsible individual, and any special considerations for the work.

work product: Any tangible item that results from a project function, activity, or task. Examples of work products include customer requirements, project plan, functional specifications, design documents, source and object code, users manuals, installation instructions, test plans, maintenance procedures, meeting minutes, schedules, budgets, and problem reports. Some subset of the work products will form the set of project deliverables.

3. SOFTWARE REENGINEERING PROJECT PLAN

This chapter provides a format and defines the contents of the software reengineering project plan. The information provided in this guide should be tailored to accommodate any existing standard for project planning. For the purposes of providing guidance on how to construct a project plan, the following structure is provided for the reader and will be used throughout the remainder of this document. The format of the project plan is structured into twelve parts:

1. STATE OBJECTIVE(S)
2. IDENTIFY BASELINE SYSTEM(S)
3. LIST ACTIVITIES
4. MAP ACTIVITIES TO PERSONNEL
5. MAP ACTIVITIES TO RESOURCES
6. SCHEDULE TIME FOR EACH ACTIVITY
7. ESTIMATE COSTS PER ACTIVITY
8. LIST RISKS WHICH COULD IMPACT PLAN
9. PLOT INFORMATION RELATIVE TO TIME
10. MARK CRITICAL SUCCESS FACTOR POINTS
11. ITERATE
12. MEASURE AND MONITOR

This structure identifies the minimum set of elements which should appear in a reengineering project plan. The following describes each part in detail, giving examples where appropriate. The related activity or activities in the Software Systems Reengineering Process Model are identified where necessary.

3.1 State Objective(s). The objectives for the software reengineering project are those project level goals which motivate the software reengineering project. One objective may be the consolidation of multiple systems or alternate configurations of the same system. This requires the integration of requirements from multiple sets of customers with varying needs. Identifying the customer is essential to insuring all personnel who may potentially be affected by the reengineering are identified and the impact on these individuals understood. It is also important to consider future user support and training on the reengineering system. Future support for the reengineered system should be outlined and preparations made to provide support to the users, including trouble reporting, corrections, and subsequent enhancement plans.

Identifying the objectives of the reengineering effort can be difficult. The activity of defining these objectives is described in the Software Systems Reengineering Process Model as *Define Objectives* [CFSW94]. Often the objectives are stated in terms that are too vague or general. Objectives that are too general make proving the success of the reengineering project very difficult. It is important that concrete and measurable objectives are identified and documented in this part of the project plan.

The objectives must identify and describe specific milestones for achieving critical success in the project. The objective statements should distinguish the goals of the various players in a reengineering project, including users, maintainers, and reengineers.

3.1.1 *Stating Objectives from Alternative Perspectives.* The reengineering project may have many objectives depending on the different people involved with the project. Three key players are the users, maintainers, and the reengineers.

The user's objectives for reengineering concern using the system and may include enhancing the user interface, changing the functional requirements, and improving the performance of the system. The known functional requirements of the existing system may be described in a user guide or other available documentation. The users of the system and the maintainers may be interviewed for defining the system's functionality from their perspective. All of the individuals who interface with the system should be consulted and their view of the systems functionality documented. The true functionality of the system, however, may not be known until the completion of the reverse engineering. It is important to document the initial perceived functionality of the system, understanding that this may not be an exact match to the true functionality as determined through the reengineering effort. The perceived functionality is initially documented from the viewpoint of both the maintainers and the users of the system.

Maintainers have objectives for supporting the system better, including decreasing maintenance costs and the ability to modify the system faster.

The reengineers may have objectives for experimenting and proving software reengineering technology concepts, including utilizing a new strategy or automated tool. The reengineered system will operate in an environment which may have certain constraints that drive its design and implementation. These environmental constraints should be understood and addressing these constraints is an objective. Target system configuration should be well-defined and any potential implications it poses addressed as an objective. Applying the appropriate standards, regulations, policy, and guidelines throughout the software reengineering effort is also an objective.

3.1.2 *How to State Concrete Objectives.* The following examples provide some insight into stating objectives in the project plan. A common objective for many software reengineering projects is the desire to "decrease maintenance costs." The objective statement would be as follows:

> "An objective of this reengineering project is to decrease maintenance costs by twenty percent."

This statement identifies the objective and also attaches a measurable quantity to the goal.

Another example is the high-level goal to "maximize customer satisfaction." An appropriate objective statement might be as follows:

> "An objective of this reengineering project is to address seventy-five percent of open critical and serious problems."

A final example is the desire to "improve software productivity." This goal could be stated as an objective in the project plan as follows:

> "An objective of this reengineering project is to decrease time spent on making each change by a minimum of fifty-percent."

3.1.3 *How to State Measurable Objectives.* The ability to focus high-level goals into measurable objectives is provided by Basili's Goal/Question/Metrics (GQM) Paradigm [Basi93]. In essence, this paradigm defines a process for turning high-level goals into concrete, measurable objectives. These objectives also require an understanding of metrics and measurements. Starting from the high-level goal, consider questions concerning this goal and what it means. From these questions it may be easier to identify metrics which quantify these questions and succeed in identifying more concrete objectives.

The GQM Paradigm is graphically represented using a tree (Figure 5) which reflects the relationship between the goals, questions, and metrics. A single metric may address more than one goal. Multiple metrics may support a single goal. There may be relevant questions which cannot be quantified, but should still be included in your objectives statements. One of the biggest mistakes in establishing goals, questions and metrics is that an organization tries to take on too many instead of narrowing the field. The key is understanding the implications these metrics have on your organization's ability to meet its goals. Decide which ones are important. Measures which cost more to measure than their value are probably not good candidates. A metric which supports more than one goal might be a good candidate. A metric which only supports one goal, but is easily measured and its implications on the organization are well-understood might be a better candidate.

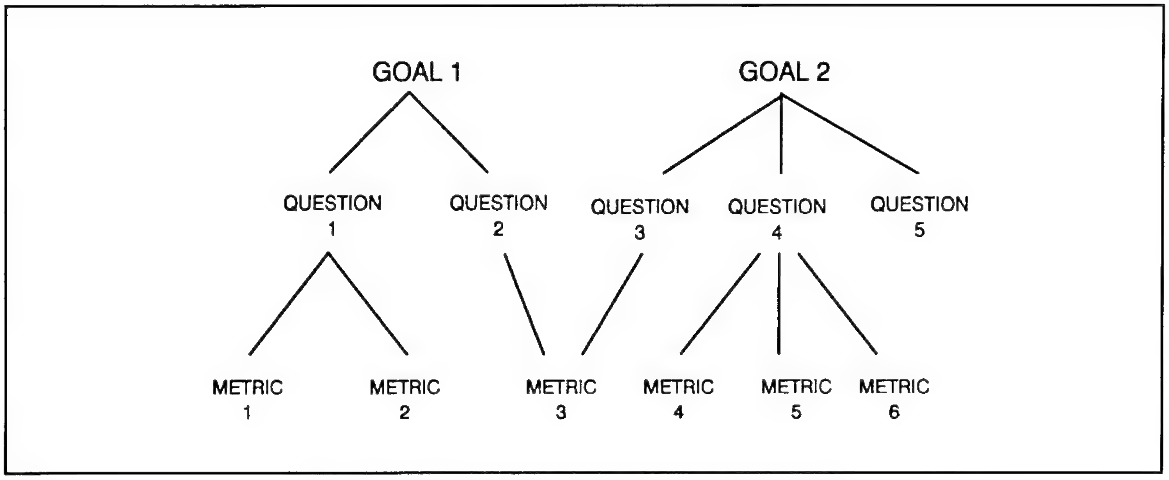


Figure 5. Goal/Question/Metric Paradigm

3.1.3 *Examples Using the G/Q/M Paradigm.* The GQM Paradigm can be applied to the three previous examples to show how measurable objectives are identified.

Increasing software productivity is a goal desired by all organizations (Figure 6). The first step in achieving this goal is establishing a process for measuring productivity in your organization. Understand what is the current productivity, including breaking down tasks into small enough jobs that each can be measured. Suggested metrics for measuring productivity include the number of non-commented source statements (NCSS) or source lines of code (SLOCs) . This metric is often measured as it relates to some unit of time, such an engineering months. Engineering months are essentially staff months, but are focused away from administrative or managerial staff months. Another measure for productivity is the number of engineering months required to perform certain phases in a project.

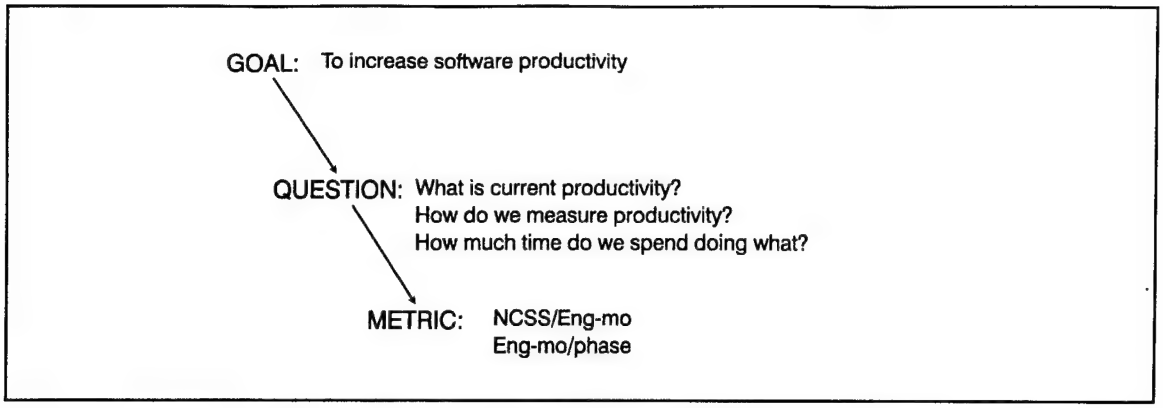

GOAL: To increase software productivity
QUESTION: What is current productivity?
How do we measure productivity?
How much time do we spend doing what?
METRIC: NCSS/Eng-mo
Eng-mo/phase

Figure 6. Example #1 Using GQM Paradigm

Maximizing customer satisfaction is another goal desired by all organizations (Figure 7). The first step in achieving this goal is establishing a process for measuring customer satisfaction in your organization. Understand whether your organization is meeting this level of customer satisfaction. Suggested metrics for measuring customer satisfaction include a raw number of open and critical problems. These problems should be categorized in some way to support you organization. The number of problems and how your organization resolves them is directly related to customer satisfaction. The problem resolution index can be used to measure the problem solving capability of your organization. This index is measured in two ways: (1) the number of problems resolved over some unit of time; and (2) the average time it takes to resolve a problem.

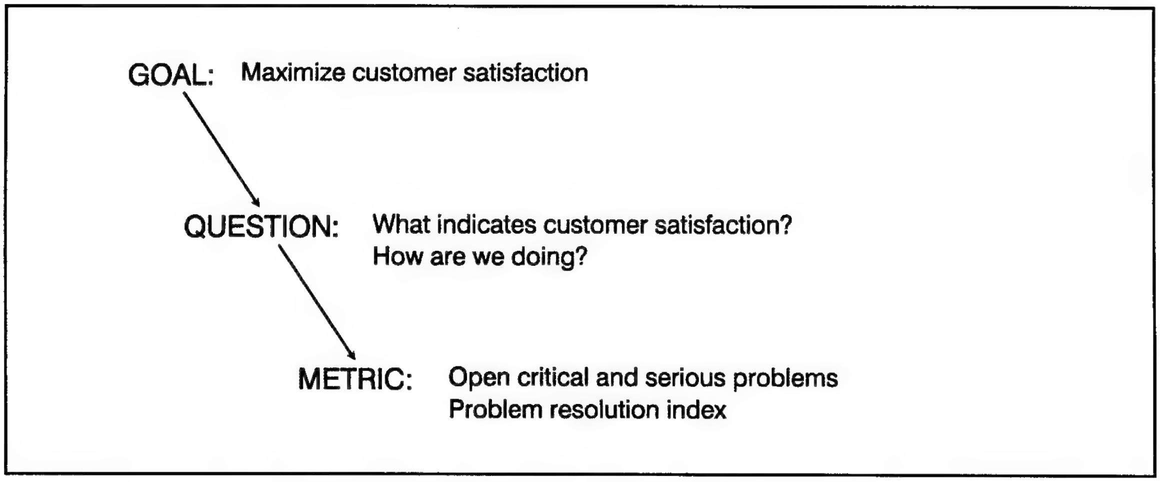

GOAL: Maximize customer satisfaction
QUESTION: What indicates customer satisfaction?
How are we doing?
METRIC: Open critical and serious problems
Problem resolution index

Figure 7. Example #2 Using GQM Paradigm

Reducing maintenance costs is probably the most important goal for any organization (Figure 8). The first step in achieving this goal is establishing a process for measuring maintenance costs in your organization. Understand what the current maintenance costs are for your organization. Suggested metrics for measuring maintenance costs include personmonths and dollars. Personnel can include labor charges for engineers, administrative assistants, and management. Dollars can include one-time costs associated with equipment purchase, material, or travel; or may be on-going charges for computer use and building space rental.

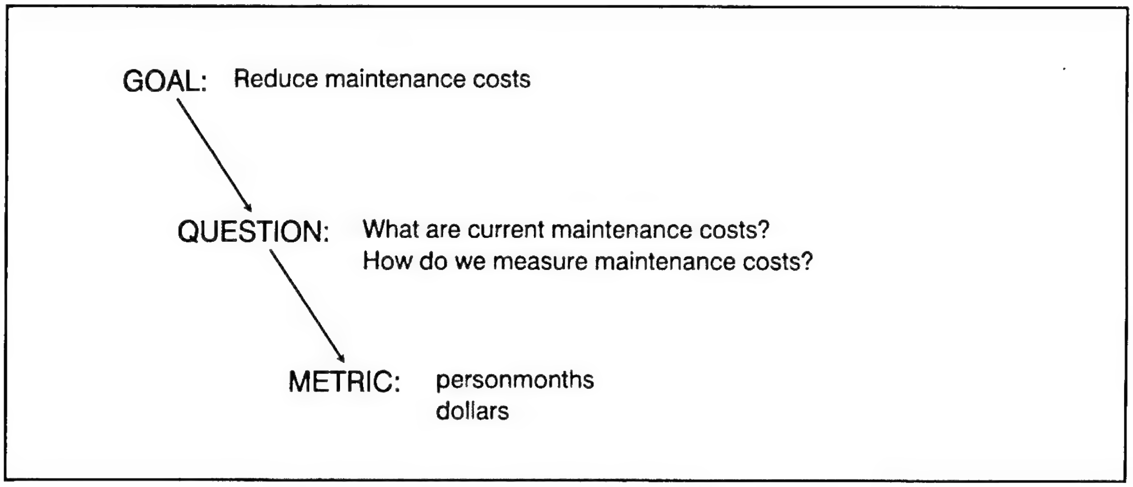

GOAL: Reduce maintenance costs
QUESTION: What are current maintenance costs?
How do we measure maintenance costs?
METRIC: personmonths
dollars

Figure 8. Example #3 Using GQM Paradigm

The GQM Paradigm is useful in identifying concrete objectives which can be mapped to measurable quantities. The ability to measure the success of the software reengineering effort starts with early planning of statements of expected results. The comparison between predicted results and achieved results at the end of the project will enable the proof of success to be established. Benchmarks of current system performance, as well as current cost of current maintenance will be measured against those of the reengineered system for improvement status.

3.2 Baseline System(s). Identify the configuration items which comprise the current automated information system. This is not an analysis activity, simply an inventory of existing system components. These items include, but are not limited to: any associated documentation, application software, data, and technical infrastructure. The activity of identifying the baseline is described in the Software Systems Reengineering Process Model as *Identify Baseline* [CFSW94].

Baselining the system is the first step in performing benchmarks. Benchmarks are an essential step in any software engineering activity for providing measures to plan and implement software system improvements. Baselining the system is also necessary to begin preparations for migrating the current system implementation to a different computer hardware platform, or integration with other software systems and commercial products.

Baselining the system means to identify existing AIS configuration items, including the application software, data, technical infrastructure, and the associated documentation.

3.3 List Activities. The high-level activities are outlined in the Software Systems Reengineering Process Model [CFSW94]. In this step, these activities should be broken down into lower level activities which can be costed and mapped to resources, including computer and personnel resources. These activities are then ordered in sequence where possible and potential parallelism identified.

3.3.1 *List Activities*. The high-level activities outlined in the Software Systems Reengineering Process Model should be further decomposed based on the particular needs of the project.

For example, *Analyze Documentation* is one of the subactivities of the activity called *Reverse Engineer*. In a sample software reengineering project the Baselined AIS Components included two user's guides, an original requirements specification document, and an electronic help system. The lower level activities were called: (1) Analyze User Guide #1, (2) Analyze User Guide #2, (3) Analyze Requirements Specification Document, (4) Analyze Electronic Help System, and (5) Integrate Analysis Results.

Another example is the breakdown of *Analyze Technical Infrastructure* into five sub-activities. These subactivities are:

(1) Analyze Existing OS Support Functions
(2) Analyze Existing Hardware Dependencies
(3) Analyze Existing External Interface Requirements
(4) Analyze Existing Commercial Components
(5) Analyze Existing Communications Requirements

The following outline lists the software reengineering activities defined in the Software Systems Reengineering Process Model. This outline is the starting point for identifying the sub-activities for an individual project.

Software Reengineering Activities

1. Reverse Engineer
    A. Analyze Documentation
    B. Analyze Application Software
    C. Analyze Data
    D. Analyze Technical Infrastructure
    E. Reconcile Extracted Products

2. Forward Engineer
    A. Analyze
    B. Design
    C. Build
    D. Integrate
    E. Test and Evaluate

When listing each activity, identify the expected product resulting from the activity [Youl90, p20]. Each activity should have one product which must be clearly defined and measurable. The minimal set of products expected from the reengineering effort includes those work products specified by the applicable DoD documentation standards. The documentation requirements specified in the MIL-STD-SDD, Military Standard for Software Development and Documentation represents a consolidation of those products specified in DoD-STD-1703(NS), Software Product Standards; DoD-STD-2167A, Defense System Software Development; and MIL-STD-7935A, DoD AIS Documentation Standards. These products are the Consolidated Data Item Descriptions (DIDs) outlined below.

Consolidated Software Design Document (C-SDD)
Consolidated Software Plan (C-SP)
Consolidated Software Requirements Document (C-SRD)
Consolidated Software Support Document (C-SSD)
Consolidated Software Test Document (C-STD)
Consolidated Software User/Operator Manual (C-SUOM)

The Consolidated DIDs are composed of individual DIDs which are also defined in the MIL-STD-SDD. For the purposes of the Software Reengineering Project Planning Guide, only the Consolidated DIDs are discussed. The individual DIDs would be broken down as defined in MIL-STD-SDD.

The following tables list the deliverables and the consumables for the software reengineering activities *Define Project* (Table 3-1), *Reverse Engineer* (Table 3-2), and *Forward Engineer* (Table 3-3). Each activity is responsible for producing these deliverables; consumables are those products which the activity needs to complete its task. The Consolidated DIDs are mapped to each of the responsible activities in the deliverables column.

Table 3-1. Deliverables and Consumables for Define Project Activities

ACTIVITY	DELIVERABLES	CONSUMABLES
DEFINE PROJECT	Consolidated Software Support Document (C-SSD) [Def.] Consolidated Software User/Operator Manual (C-SUOM) [Def.] Baselined AIS Components Reengineering Project Plan	Business Requirements AIS Feasibility Analysis Results Reverse Engineered Products Analysis Deliverables
Define Objectives	Objectives	Business Requirements AIS Feasibility Analysis Results Recommended Changes to Objectives
Identify Baseline	Baselined AIS Components - Existing Software - Existing Data - Existing Technical Infrastructure	Business Requirements AIS Feasibility Analysis Results Reverse Engineered Products Analysis Deliverables Recommended Changes to Baseline
Define Reengineering Project Plan	Reengineering Project Plan - Reengineering Project Strategy - Proposed Methodologies and Tools - Project Resources	Baselined AIS Components Objectives Analysis Deliverables Reverse Engineered Products

Table 3-2. Deliverables and Consumables for Reverse Engineer Activities

ACTIVITY	DELIVERABLES	CONSUMABLES
REVERSE ENGINEER	Consolidated Software Design Document (C-SDD) Consolidated Software Plan (C-SP) Consolidated Software Requirements Document (C-SRD) Consolidated Software Support Document (C-SSD) Consolidated Software User/Operator Manual (C-SUOM) Candidate Reuse Assets Reverse Engineered Products	Business Requirements Reusable Assets Baselined AIS Components
Analyze Documentation	Extracted Documentation Products	Baselined AIS Components Recommended Product Revisions
Analyze Application Software	Extracted Software Products	Reusable Assets Baselined AIS Components Recommended Product Revisions
Analyze Data	Extracted Data Products	Reusable Assets Baselined AIS Components Recommended Product Revisions
Analyze Technical Infrastructure	Extracted Technical Infrastructure Products	Baselined AIS Components Recommended Product Revisions
Reconcile Extracted Products	Candidate Reuse Assets Reverse Engineered Products Recommended Product Revisions	Business Requirements Extracted Software Products Extracted Documentation Products Extracted Technical Infrastructure Products Extracted Data Products

Table 3-3. Deliverables and Consumables for Forward Engineer Activities

ACTIVITY	DELIVERABLES	CONSUMABLES
FORWARD ENGINEER	Consolidated Software Support Document (C-SSD) [Rev.] Consolidated Software Test Document (C-STD) Consolidated Software User/Operator Manual (C-SUOM) Reengineered System Candidate Reuse Assets	Business Requirements Reverse Engineered Products Reusable Assets
Analyze	Consolidated Software Requirements Document (C-SRD) Consolidated Software Test Document (C-STD) [Def.] - Analysis Deliverables	Business Requirements Reverse Engineered Products Reusable Assets Design Results Test Results
Design	Consolidated Software Design Document (C-SDD) [Rev.] Consolidated Software Plan (C-SP) [Rev.] - Design Components Design Results	Reverse Engineered Products Reusable Assets Build Results Analysis Deliverables
Build	Consolidated Software User/Operator Manual (C-SUOM) [Rev.] Build Components Build Results	Reusable Assets Design Components Integration Components Test Results
Integrate	Integration Components Integration Results	Reusable Assets Build Components Test Results
Test and Evaluate	Consolidated Software Test Document (C-STD) [Rev.] - Test Results	Analysis Deliverables Reusable Assets Build Components Integration Components

The tables on the previous pages also provide a template for listing the expected products from each of the lower-level software reengineering activities in the project. Each lower-level activities should be added to the tables and their expected products listed.

3.3.2 *Order Activities in Sequence.* The Software Systems Reengineering Process Model outlines the high-level activities of software reengineering. Each individual project may implement all or part of these activities. The activities and their relationships with each other are not related to, concerned with, or limited by time. Activities may be dependent: one activity requires the product of another activity to complete its product. Other activities may have no relationship and can potentially be performed in parallel. This step orders the activities in a sequence to determine a time relation for the lower-level activities.

3.4 Map Activities To Personnel. This step assigns the appropriate personnel to each activity. More than one person may be required to perform the activity. How individuals work together may influence how work teams are identified and structured. Consideration should be given to personnel experience and skill in the activities, tools, and the system components involved. Proximity of people to one another may impact their ability to perform the activity.

The skills of personnel are improved through training. All personnel should be trained in the latest advances in modern software engineering principles supported by processes, tools, and languages. Experience provides personnel with the knowledge of the current system environment, but training is needed to insure a completed understanding of the target environment for supporting the reengineered system and the technology available for migrating to that environment.

3.5 Map Activities To Resources. Assign resources required for performing each activity. Available resources for performing the software reengineering may be limited to those currently available in the organization. These resources are mapped to the activity which they best support. These resources include personnel, computer, reusable components, and available technology for supporting the software reengineering effort.

The experience and skill of the personnel utilizing these tools should complement the tools capabilities to automate the activity. The technical complexity of the tools also impacts the personnel's ability to use the tool and implement the activities. The tools for completing the activity must be available to the personnel. The correct platform and documentation for using these tools must be readily available.

Modern software engineering is supported through a variety of methodologies and tools which were not available when the existing engineering environments in most organizations were established. It is important that alternative technologies be examined for their applicability in the organization's business process. A plan for integrating these technologies into the organization should be established which includes training for all effected personnel.

3.6 Schedule Time For Each Activity. Estimate the duration of each activity in units of time that can be costed and staffed. Many of the high-level activities in software reengineering have never been performed in most organizations. However, these activities are broken down into the lowest level activities which may be compatible with simple software engineering activities. The time for performing these activities is comparable with many system support activities. Experience in each individual organization should be used for scheduling time for these software reengineering activities.

3.7 Estimate Costs Per Activity. For each activity estimate the cost and budget. The lowest level software reengineering activities which were broken down and compared to software engineering activities in Step 6 of this Guide can also be used to estimate the costs. The cost for these activities is available from current system support experience in each individual organization and should be used for estimating cost for the software reengineering activities.

3.8 List Risks Which Could Impact Plan. For each activity list the issues which could potentially impact the performance of each activity. For example, list the factors which could alter both the start and completion date of that activity. Identifying the potential problems ahead enables the manager to initiate preemptive action. Alternative scenarios based on these problems should be outlined. There are many types of risks including the following [Youl90, p25]:

- hardware or software problems
- customer requirement inconsistencies
- key staff may leave
- application pitfalls
- external suppliers
- availability of staff
- inexperienced staff in modern software engineering principles, tools/languages

The Software Reengineering Risk Taxonomy [CIM93c] assists project managers in identifying potential hazards when performing software reengineering.

While risk identification is key to a project plan, managing the risks when they become a reality can be very difficult. Planning to manage these risks is essential to mitigating the effect on the success of the project. Steps can be posed for avoiding risks once they are identified. There are several sources for managing risk, including the following steps outlined by B. Boehm in *Software Risk Management* [Boeh89].

I. Risk Management

- A. Risk Assessment
  - 1. Risk Identification
    - a. checklists
    - b. decision driver analysis
    - c. assumption analysis
    - d. decomposition
  - 2. Risk Analysis
    - a. performance models
    - b. cost models
    - c. network analysis
    - d. decision analysis
    - e. quality factor analysis
  - 3. Risk Prioritization
    - a. risk exposure
    - b. risk leverage
    - c. compound reduction

- B. Risk Control
  - 1. Risk Management Planning
    - a. buying information
    - b. risk avoidance
    - c. risk transfer
    - d. risk reduction
    - e. risk element planning
    - f. risk plan integration
  - 2. Risk Resolution
    - a. prototypes
    - b. simulations
    - c. benchmarks
    - d. analyses
    - e. staffing
  - 3. Risk Monitoring
    - a. milestone tracking
    - b. top-10 tracking
    - c. risk reassessment
    - d. corrective action

Three sample risk areas include introducing new technology, customer requirement inconsistencies, and the inexperience of staff in principles/tools/languages. The following examples show how a risk area is identified and the impact described in the project plan. Plans for mitigating each risk identified in the project plan should be explored and also stated in the project plan. Notice the statement describing the risk mitigator in each of these examples.

Example Risk Area #1: Introducing New Technology:

Example Risk Statement: "This reengineering project requires that the reengineered system reside in a client/server environment. Personnel have no experience in client/server technology. The impact of moving to this type of environment is unknown at this time.

Risk Mitigator: Training in client/server technology is suggested for all project team members. Reverse engineering will analyze the current hardware platform capabilities and forward engineering will compare these capabilities to a client/server environment."

Example Risk Area #2: Customer Requirement Iconsistencies:

Example Risk Statement: "Current customer requirements are not being met due to incompatible requirements from multiple customers, unfulfilled change requests, and perceived unknown requirements. The inability to meet customer requirements may result in customers abandoning this system.

Risk Mitigator: A joint application design/development (JAD) approach is suggested with representatives from customers, maintainers, and reengineering project personnel."

Example Risk Area #3: Inexperience of staff :

Example Risk Statement: "Current staff has no formal training or experience with the Ada programming language. Without experience there will be a risk is generating quality Ada code manually and an inability to adequately evaluate the automatically generated Ada code. Without training this risk is even greater.

Risk Mitigator: Ada training for 6-8 weeks is suggested for all affected personnel."

3.9 Plot Information Relative To Time. The schedule, mapping of personnel, and mapping of resources can be graphed using Gantt charts, a popular technique for representing project management information. The following is a sample Gantt chart for the high-level activities within software reengineering. The lower-level activities should be added to this chart for each individual project.

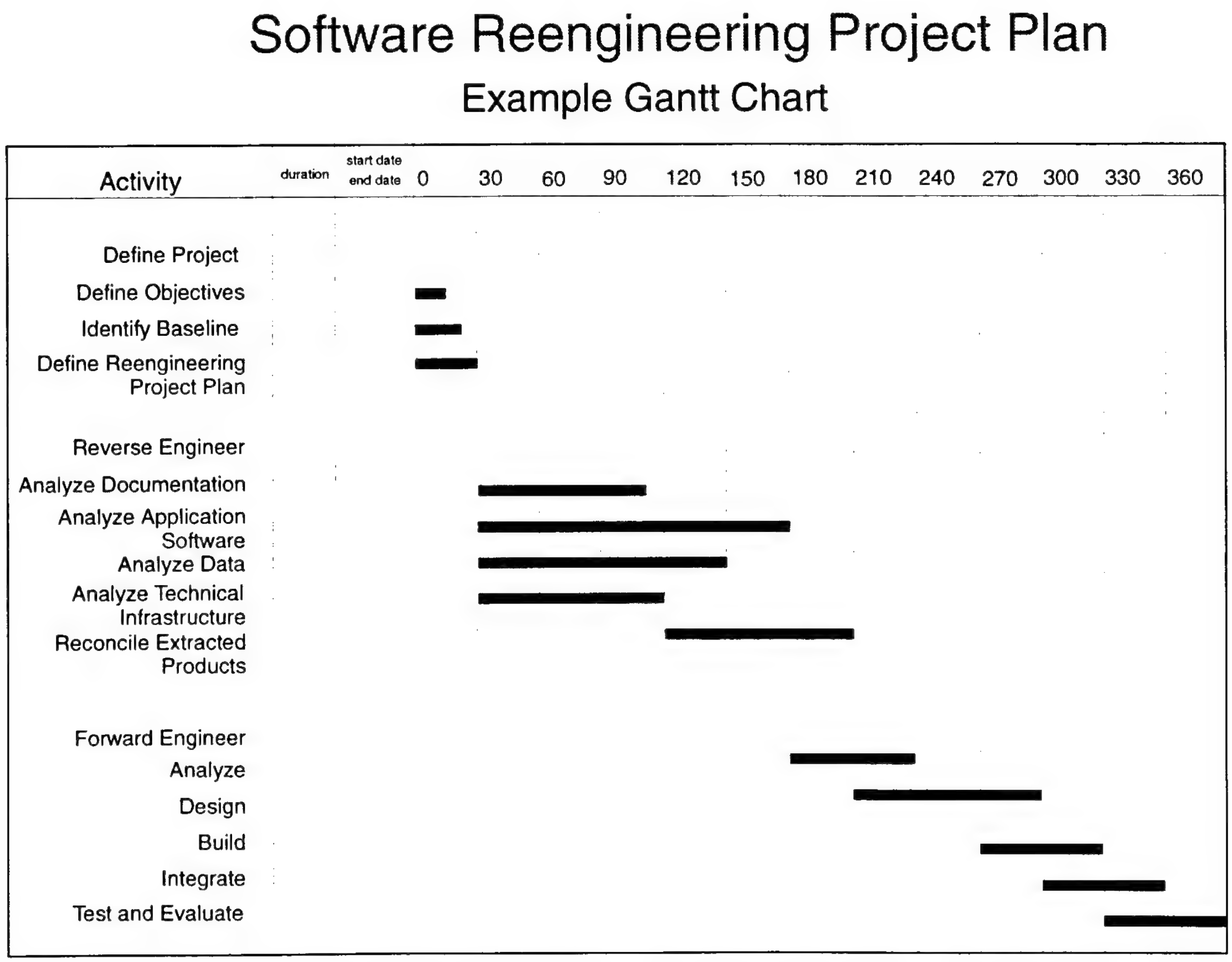


Figure 9. Sample Gantt Chart for Software Reengineering Project Planning

3.10 Mark Critical Success Factors On Timeline. List the critical points in the software reengineering project which may indicate the success of the project. These are essential to monitoring the progress of the project. Using the timeline, note these points on the schedule. On a separate page for each point describe the status of the project at that point, including costs, products, and other indicators of progress.

Measures of success were outlined as part of Step 1 in this Guide. In this step, these measures are considered in detail. Minimal deviations from the schedule and acceptable cost differentiations should be identified. Overall performance improvements and qualitative advancements should also be identified. Benchmarks of the current system support environment and performance are necessary for comparison to measures taken throughout and at the end of the software reengineering effort.

A successful software reengineering effort has several attributes. These attributes are often not measurable until the completion of the effort. Identifying the critical success factors throughout the duration of the project allows for modifications to be made to the project plan and increase the likelihood of success.

The following example identifies the objective and its related critical success factor (CSF) which would be included in the project plan.

If your objective was: "An objective of this reengineering project is to decrease time spent on making each change by a minimum of fifty-percent."

Then, a CS would be: "An increase in software productivity of a minimum of twenty-five percent is necessary to achieve critical success."

3.11 Plan to Capture Lessons Learned. Many of the activities identified throughout the first steps in this Guide are continuous activities which should be performed to constantly improve the software products. By performing these activities repeatedly and improving the product, the process itself can be improved. Individual needs of each organization can be addressed through subtle improvements in the process. Use the experience the establish process improvement goals. Outline a framework for achieving these goals in this section of the project plan.

This section of the project plan should define a process for capturing lessons learned. It is not enough to state in the project plan that this will be done. A specific plan on how this will be accomplished must be described in this part of the plan. For example, establish a biweekly report that identifies key lessons learned during that phase of the project. In this part of the project plan include a copy of the report template which will be used by the project team. Mark on the time line when these reports are due.

This part of the project plan should also identify other projects in the organization which are potential candidates for benefitting from the experience gained during this project. Establish a mechanism for performing technology transition within the organization.

3.12 Plan to Measure and Monitor. Methods and guides for measuring both products and process are defined in several sources. The DoD defined a set of Core Measures which provide fundamental information for project planning, project management, and software process improvement. These measures include size, effort, and schedule. The Center for Software's Metrics Program is examining how these core measures impact software reengineering and reuse activities. The Metrics Program is also directing a number of DoD organizations in the collection and analysis of these measures. Monitoring these sites and other activities supports the development of a Metrics Architecture and the initiation of a DoD-wide metrics program.

REFERENCES

[Basi93] V.R. Basili, *Software Modeling and Measurement: The Goal/Question/Metric Paradigm,* Institute for Advanced Computer Studies, Department of Computer Science, University of Maryland, developed under NASA/GSFC contract NSG-5123 and AFOSR contract 90-0031, 1993.

[Blum92] B.I. Blum, *Software Engineering: A Hollistic View*, Oxford University Press, 1992.

[Boeh89] B. W. Boehm, *Software Risk Management*, IEEE Computer Society Press, Washington, DC, 1989.

[CFSW94] Software Systems Reengineering Process Model: Version 2.0, Center for Software, September 1994.

[CIM93a] Information System Criteria for Applying Software Reengineering, Center for Information Management, May 1993.

[CIM93b] Software Systems Reengineering Process Model: Version 1.0, Center for Information Management, August 1993.

[CIM93c] Software Reengineering Risk Taxonomy, Center for Information Management, September 1993.

[CMU92] Software Measurement for DoD Systems: Recommendations for Initial Core Measures, TR CMU/SEI-92-TR-19, September 1992.

[DSD93] Deputy Secretary of Defense memorandum, October 13, 1993, subj: "Accelerated Implementation of Migration Systems, Data Standards, and Process Improvement."

[Hump89] W.S. Humphrey, *Managing the Software Process*, Software Engineering Institute, Addison-Wesley Publishing Company, 1989.

[Pres87] R. S. Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill, Inc., Second Edition, 1987.

[Whit90] N. Whitten, *Managing Software Development Projects*, John Wiley and Sons, Inc., 1990.

[Youl90] D.P. Youll, *Making Software Development Visible: Effective Project Control*, John Wiley and Sons, Inc., 1990

Appendix A
Project Planning Templates

The following templates have been prepared to aid in the documentation of the information required in a Software Reengineering Project Plan as defined in this guide. These templates should be used to outline the initial project plan for any software reengineering effort. Each step in the plan which is outlined in Section 3.0 of the Software Reengineering Project Planning Guide, Version 2.0 corresponds to a set of templates on the following pages. For example, Section 3.1 in the Guide corresponds to Step 1. State Objectives. Please refer to the Guide for additional information on how to use these templates. These pages can be pulled out of this appendix and duplicated for use in project planning.

(This page was intentionally left blank.)

Software Reengineering Project Plan

Sponsoring Organization: ______________________
Project Name: ______________________

This project plan is composed of the following twelve parts:

1. STATE OBJECTIVE(S)
2. IDENTIFY BASELINE SYSTEM(S)
3. LIST ACTIVITIES
4. MAP ACTIVITIES TO PERSONNEL
5. MAP ACTIVITIES TO RESOURCES
6. SCHEDULE TIME FOR EACH ACTIVITY
7. ESTIMATE COSTS PER ACTIVITY
8. LIST RISKS WHICH COULD IMPACT PLAN
9. PLOT INFORMATION RELATIVE TO TIME
10. MARK CRITICAL SUCCESS FACTOR POINTS
11. PLAN TO CAPTURE LESSONS LEARNED
12. PLAN TO MEASURE AND MONITOR

1. State Objective(s). The objectives for the software reengineering project may include those objectives for using the system, supporting the system, and those objectives for utilizing software reengineering technology. The activity of defining these objectives is described in the Software Systems Reengineering Process Model as *Define Objectives* [CIM93b]. {The following template provides a form for documenting the information required in this part of the project plan:}

STATE OBJECTIVES

Objective #1:
Viewpoint:
Issues:
Metric(s):

Objective #2:
Viewpoint:
Issues:
Metric(s):

2. Baseline System(s). The following configuration items comprise the current automated information system. This is not an analysis of these items, only an inventory of existing system components. These items include, but are not limited to: any associated documentation, application software, data, and technical infrastructure. The activity of identifying the baseline is described in the Software Systems Reengineering Process Model as *Identify Baseline* [CIM93b]. {The following text supplements the description of this activity in the Process Model.}

BASELINE IDENTIFICATION

Identify Existing Application Software:
Name of software item:
Language:
Size:
Description:

Associated Documentation:

Identify Existing Data:
Name of each data element:
Description:

Associated Documentation:

Identify Existing Technical Infrastructure:
Names of components:
Description:

Associated Documentation:

(This page was intentionally left blank.)

3. List Activities. The high-level activities are outlined in the Software Systems Reengineering Process Model [CIM93b]. In this step, these activities should be broken down into lower level activities which can be costed and mapped to resources, including computer and personnel resources. These activities are then ordered in sequence where possible and potential parallelism identified. These activities form the basis of the reengineering strategy as described in the Software Systems Reengineering Process Model as *Define Strategy* [CIM93b]. {The following provides a template for outlining the lower-level software reengineering activities in the project:}

LIST ACTIVITIES

1. Reverse Engineer

   A. Analyze Documentation
      1.
      2.
      3.
      :

   B. Analyze Application Software
      1.
      2.
      3.
      :

   C. Analyze Data
      1.
      2.
      3.
      :

   D. Analyze Technical Infrastructure
      1.
      2.
      3.
      :

   E. Reconcile Extracted Products
      1.
      2.
      3.
      :

LIST ACTIVITIES (cont.)

2. Forward Engineer

   A. Analyze
      1.
      2.
      3.
      :

   B. Design
      1.
      2.
      3.
      :

   C. Build
      1.
      2.
      3.
      :

   D. Integrate
      1.
      2.
      3.
      :

   E. Test and Evaluate
      1.
      2.
      3.
      :

(This page was intentionally left blank.)

4. Map Activities To Personnel. This step assigns the appropriate personnel to each activity. More than one person may be required to perform the activity. The activity of mapping activities to personnel is described in the Software Systems Reengineering Process Model as *Allocate Resources* [CIM93b]. {The following provides a template for mapping personnel to the lower-level software reengineering activities in the project:}

MAP ACTIVITIES TO PERSONNEL

1. Reverse Engineer — Personnel:

A. Analyze Documentation

1.
2.
3.
:

B. Analyze Application Software

1.
2.
3.
:

C. Analyze Data

1.
2.
3.
:

D. Analyze Technical Infrastructure

1.
2.
3.
:

E. Reconcile Extracted Products

1.
2.
3.
:

MAP ACTIVITIES TO PERSONNEL (cont.)

2. Forward Engineer Personnel:

A. Analyze

1.
2.
3.
:

B. Design

1.
2.
3.
:

C. Build

1.
2.
3.
:

D. Integrate

1.
2.
3.
:

E. Test and Evaluate

1.
2.
3.
:

(This page intentionally left blank.)

5. Map Activities To Resources. Assign resources required for performing each activity. Available resources for performing the software reengineering may be limited to those currently available in the organization. These resources are mapped to the activity which they best support. These resources include computer resources, reusable components, and available technology for supporting the software reengineering effort. The activity of mapping activities to resources is described in the Software Systems Reengineering Process Model as *Allocate Resources* [CIM93b]. {The following provides a template for mapping resources to the lower-level software reengineering activities in the project:}

5. MAP ACTIVITIES TO RESOURCES

1. Reverse Engineer Resource: Quantity:

A. Analyze Documentation
1.
2.
3.
:

B. Analyze Application Software
1.
2.
3.
:

C. Analyze Data
1.
2.
3.
:

D. Analyze Technical Infrastructure
1.
2.
3.
:

E. Reconcile Extracted Products
1.
2.
3.
:

5. MAP ACTIVITIES TO RESOURCES (cont.)

2. Forward Engineer	Resource:	Quantity:

A. Analyze

1.
2.
3.
:

B. Design

1.
2.
3.
:

C. Build

1.
2.
3.
:

D. Integrate

1.
2.
3.
:

E. Test and Evaluate

1.
2.
3.
:

(This page intentionally left blank.)

6. Schedule Time For Each Activity. Estimate the duration of each activity in units of time that can be costed and staffed. Many of the high-level activities in software reengineering have never been performed in most organizations. However, these activities are broken down into the lowest level activities which may be compatible with simple software engineering activities. The time for performing these activities is comparable with many system support activities. Experience in each individual organization should be used for scheduling time for these software reengineering activities. The activity of scheduling time for each activity is performed in the Software Systems Reengineering Process Model in *Define Reengineering Project Plan* [CIM93b]. {The following template provides a form for documenting the information required in this part of the project plan:}

6. SCHEDULE TIME FOR EACH ACTIVITY

1. Reverse Engineer Time(Days/Weeks/Months):

A. Analyze Documentation
1.
2.
3.
:

B. Analyze Application Software
1.
2.
3.
:

C. Analyze Data
1.
2.
3.
:

D. Analyze Technical Infrastructure
1.
2.
3.
:

E. Reconcile Extracted Products
1.
2.
3.
:

6. SCHEDULE TIME FOR EACH ACTIVITY (cont.)

2. Forward Engineer Time(Days/Weeks/Months):

A. Analyze

1.
2.
3.
:

B. Design

1.
2.
3.
:

C. Build

1.
2.
3.
:

D. Integrate

1.
2.
3.
:

E. Test and Evaluate

1.
2.
3.
:

(This page intentionally left blank.)

7. Estimate Costs Per Activity. For each activity estimate the cost and budget. The lowest level software reengineering activities which were broken down and compared to software engineering activities in Step 6 of this Guide can also be used to estimate the costs. The cost for these activities is available from current system support experience in each individual organization and should be used for estimating cost for the software reengineering activities. The activity of estimating costs is performed in the Software Systems Reengineering Process Model in the activity *Define Reengineering Project Plan* [CIM93b]. {The following template provides a form for documenting the information required in this part of the project plan:}

7. ESTIMATE COSTS PER ACTIVITY

1. Reverse Engineer Estimated Cost:

   A. Analyze Documentation
      1.
      2.
      3.
      :

   B. Analyze Application Software
      1.
      2.
      3.
      :

   C. Analyze Data
      1.
      2.
      3.
      :

   D. Analyze Technical Infrastructure
      1.
      2.
      3.
      :

   E. Reconcile Extracted Products
      1.
      2.
      3.
      :

7. <u>ESTIMATE COSTS PER ACTIVITY</u> (cont.)

2. Forward Engineer Estimated Cost:

A. Analyze
   1.
   2.
   3.
   :

B. Design
   1.
   2.
   3.
   :

C. Build
   1.
   2.
   3.
   :

D. Integrate
   1.
   2.
   3.
   :

E. Test and Evaluate
   1.
   2.
   3.
   :

8. List Risks Which Could Impact Plan. For each activity list the issues which could potentially impact the performance of each activity. For example, list the factors which could alter both the start and completion date of that activity. Identifying the potential problems ahead enables the manager to initiate preemptive action. Alternative scenarios based on these problems should be outlined. The activity of identifying risks is performed in the Software Systems Reengineering Process Model in the activity *Define Reengineering Project Plan* [CIM93b]. {The following template provides a form for documenting the information required in this part of the project plan:}

8. LIST RISKS WHICH COULD IMPACT PLAN

Risk Area:

Risk Statement:

Risk Mitigator:

Risk Area:

Risk Statement:

Risk Mitigator:

9. Plot Information Relative To Time. The schedule, mapping of personnel, and mapping of resources can be graphed using Gantt charts, a popular technique for representing project management information. The following is a sample Gantt chart for the high-level activities within software reengineering. The lower-level activities should be added to this chart for each individual project.

9. PLOT INFORMATION RELATIVE TO TIME

Software Reengineering Project Plan

Project Name: ____________________

Activity	duration	start date end date	0	30	60	90	120	150	180	210	240	270	300	330	360
Define Project															
Define Objectives															
Identify Baseline															
Define Reengineering Project Plan															

Software Reengineering Project Plan

Project Name: ____________________

Activity	duration	start date end date	0	30	60	90	120	150	180	210	240	270	300	330	360
Reverse Engineer															
Analyze Documentation															
Analyze Application Software															
Analyze Data															
Analyze Technical Infrastructure															
Reconcile Extracted Products															

Software Reengineering Project Plan

Project Name: ____________________

Activity	duration	start date end date	0	30	60	90	120	150	180	210	240	270	300	330	360
Forward Engineer															
Analyze															
Design															
Build															
Integrate															
Test and Evaluate															

10. Mark Critical Success Factors On Timeline. List the critical points in the software reengineering project which may indicate the success of the project. These are essential to monitoring the progress of the project. Using the timeline, note these points on the schedule. On a separate page for each point describe the status of the project at that point, including costs, products, and other indicators of progress.

10. MARK CRITICAL SUCCESS FACTORS ON TIMELINE

Related Objective:
Critical Success Factor:
Scheduled Completion Date:

Related Objective:
Critical Success Factor:
Scheduled Completion Date:

Related Objective:
Critical Success Factor:
Scheduled Completion Date:

11. Plan to Capture Lessons Learned. Many of the activities identified throughout the first steps in this Guide are continuous activities which should be performed to constantly improve the software products. By performing these activities repeatedly and improving the product, the process itself can be improved. Individual needs of each organization can be addressed through subtle improvements in the process.

Outline a plan to incorporate the lessons learned from this project into other aspects of the organization's work:

11. PLAN TO CAPTURE LESSONS LEARNED

12. Plan to Measure and Monitor. Methods and guides for measuring both products and process are defined in several sources. The Software Engineering Institute (SEI) has defined a set of core measures to provide fundamental information for project planning, project management, and software process improvement [CMU92]. These measures are size, effort, defects, and schedule. The Center for Software's Metrics Program is examining how these core measures impact software reengineering and reuse activities. The Metrics Program is also directing a number of DoD organizations in the collection and analysis of these measures. Monitoring these sites and other activities support the development of a Metrics Architecture and the initiation of a DoD-wide metrics program.

Outline a plan to continue measuring and improving the software engineering environment:

12. PLAN TO MEASURE AND MONITOR